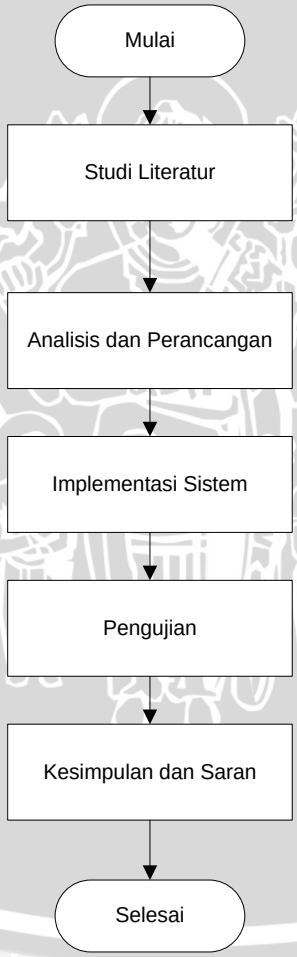
BAB III
METODE PENELITIAN DAN PERANCANGAN

Pada bab ini dijelaskan langkah-langkah yang akan ditempuh dalam menyusun skripsi, yaitu studi literatur, analisis perancangan, implementasi, pengujian sistem, serta kesimpulan dan saran disertakan sebagai catatan dan kemungkinan arah untuk pengembangan selanjutnya. Urutan langkah-langkah yang akan dilakukan pada penelitian digambarkan pada blok diagram sebagai berikut :



Gambar 3.1 Diagram alir metode penelitian

Sumber : [Rancangan]

Adapun penjelasan mengenai diagram alir metode penelitian adalah sebagai berikut ;

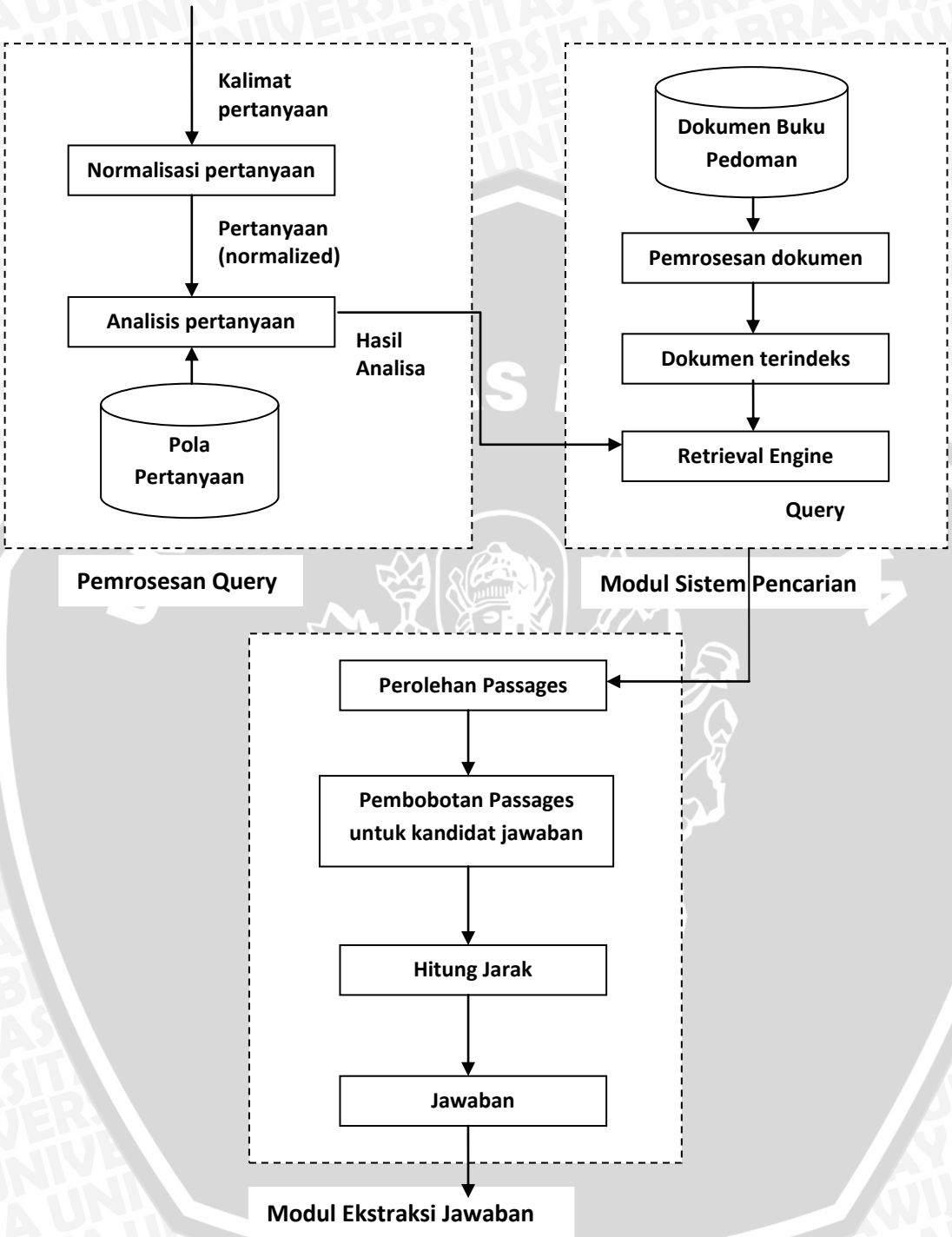
1. Mempelajari konsep preprocessing pada text mining, Algoritma Rabin-Karp dalam pencocokan string dan pembobotan heuristic, serta Algoritma Rule-Based pada Question Answering System.
2. Melakukan analisis dan perancangan sistem untuk implementasi Question Answering System.
3. Melakukan implementasi sistem berdasarkan analisis dan perancangan sistem yang telah dilakukan.
4. Melakukan uji coba terhadap sistem, dengan membandingkan jawaban yang relevan menggunakan pembobotan Rabin-Karp dan pembobotan Rule-Based.
5. Mengevaluasi hasil uji coba.

3.1 Studi Literatur

Penulis melakukan studi literatur mengenai dasar teori yang digunakan untuk menunjang penulisan skripsi yang berkaitan dengan *Question Answering System*, *stemming* Arifin-Setiono, algoritma Rabin-Karp, algoritma *Rule-Based*, dan pembobotan heuristik pada *Question Answering System* serta uji akurasi. Literatur yang diperoleh kemudian diringkas dan dijelaskan pada bab 2.

3.2 Analisa Sistem

Pada pembuatan aplikasi *Question Answering System* proses pembelajaran sistem selama ini dilakukan secara manual, yakni pertanyaan dan jawabannya telah tersedia dalam *database*. Oleh karena itu pada penelitian ini diterapkan algoritma untuk memaksimalkan pemberian jawaban yang relevan tehadap pertanyaan yang diberikan secara lebih mudah dan cepat dibandingkan sistem manual. Dari studi literatur yang telah dilakukan, penulis kemudian melakukan analisa sistem. Analisis sistem bertujuan untuk mendapatkan semua kebutuhan yang diperlukan dari sistem yang akan dibangun. Berikut ini adalah skema arsitektur *Question Answering System* yang menjelaskan jalannya sistem.



Gambar 3.2 Arsitektur *Question Answering System*

Sumber : [Rancangan]

Sistem yang akan dibuat digunakan untuk implementasi *Question Answering System*, adapun prosesnya dimulai dari input pertanyaan oleh user. Pada modul Pemrosesan *Query* kalimat pertanyaan dinormalisasi dengan melakukan *preprocessing*. Tahap *preprocessing* yang dilakukan pada sistem ini adalah *tokenizing*, *case fold*ing, *filtering* (penghilangan kata-kata yang tidak penting), dan *stemming*. Tahapan *preprocessing* akan menghasilkan kumpulan kata-kata penting (*keyword*) dalam bentuk kata dasarnya. Kalimat pertanyaan juga dianalisa berdasarkan tipe pertanyaan untuk mendapatkan pola pertanyaan, hal ini dimaksudkan untuk mendapatkan kandidat jawaban berdasarkan *named entity* pada pola pertanyaan.

Hasil Pemrosesan *Query* disimpan dalam *Retrieval Engine* berupa *keyword* untuk proses ekstraksi jawaban. Kemudian dilakukan pencarian pada dokumen menggunakan *Search Engine* sesuai *keyword* pada kalimat pertanyaan .Tahap *preprocessing* juga dilakukan pada dokumen teks Buku Pedoman Pendidikan Program Teknologi Informasi dan Ilmu Komputer Universitas Brawijaya Tahun Akademik 2012/2013. Dokumen-dokumen yang diperoleh akan menjadi kandidat jawaban dan disimpan dalam *Retrieval Engine*.

Pada modul ekstraksi jawaban dilakukan pembobotan terhadap *passages* dengan pembobotan Rabin-Karp dan Rule-Based. Pada pembobotan Rabin-Karp menggunakan metode *heuristic*, dengan membentuk pola kata dan mengubahnya dalam bentuk *hash value* dari *passages*, maka akan diketahui *count query* (jumlah kata yang terdapat pada *query* / kalimat tanya), *count match* (jumlah hasil pencocokan antara kata yang terdapat pada *query* dan *passage*), dan nilai lainnya. Kandidat jawaban yang memiliki jarak kurang dari *threshold* dipilih menjadi entitas jawaban terakhir. Pada pembobotan Rule-Based diterapkan fungsi *Wordmatch* yang merupakan nilai perbandingan antara kalimat *query* dengan kalimat pada dokumen, kemudian diterapkan Algoritma Rule-Based yang terdapat pada penelitian Riloff dan Thelen (2000) serta penelitian Ikhsani (2006). Kemudian dipilih jawaban berupa kalimat pada dokumen yang memiliki nilai skor tertinggi.

3.2.1 Kebutuhan *User Interface*

Untuk mengimplementasikan sistem *Question Answering* pada program aplikasi dibutuhkan perancangan awal *user interface* (antar muka) yang meliputi : Perancangan Input dan Perancangan Prototype pada sistem.

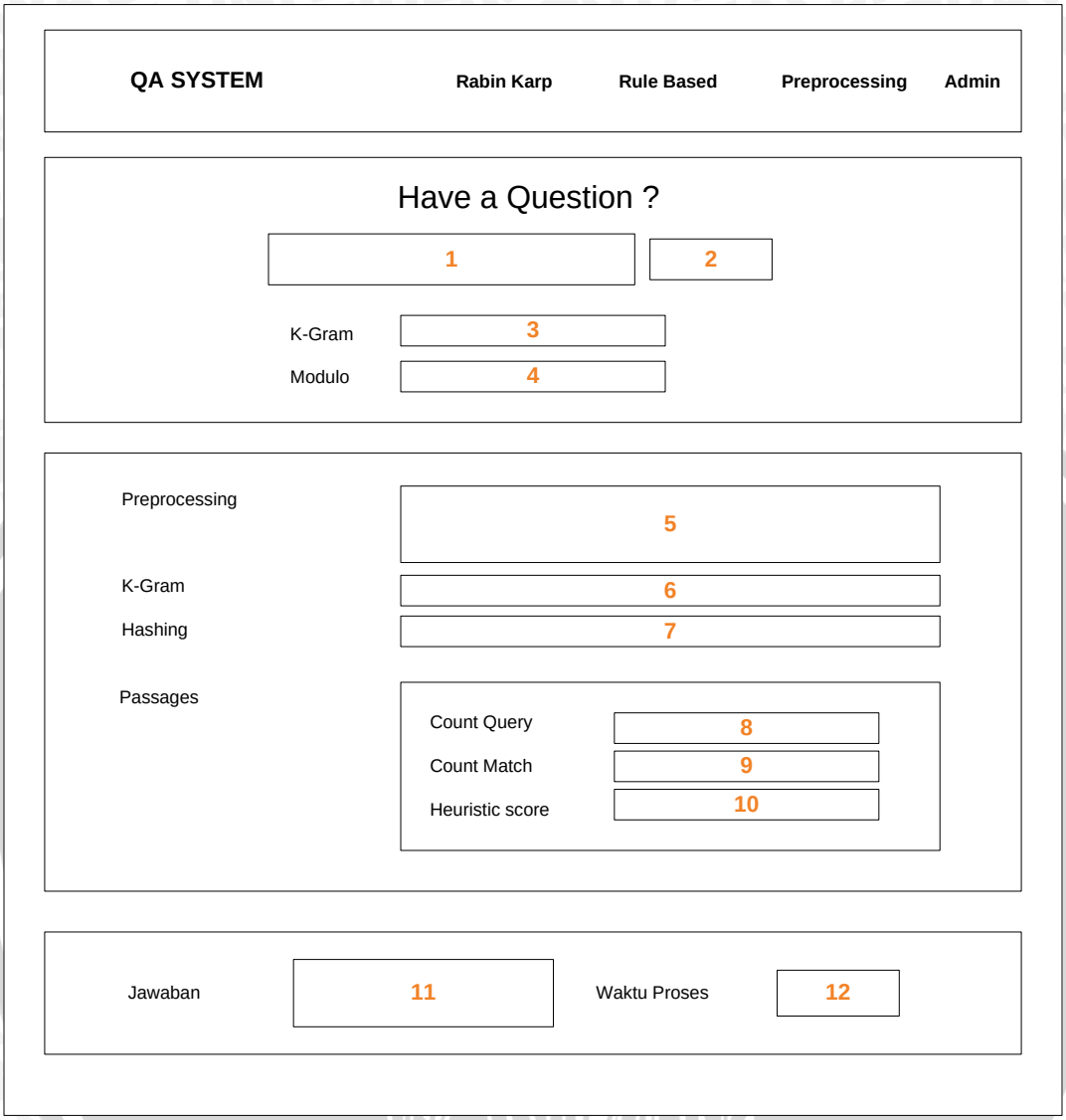
3.2.1.1 Perancangan Input pada Sistem

Pada implementasi *Question Answering System* ini ada beberapa inputan dan parameter yang harus diisi oleh user untuk diproses, yaitu :

1. Algoritma yang akan digunakan untuk proses ekstraksi jawaban. (pembobotan heuristik *Rabin-Karp* atau pembobotan *Rule-Based*).
2. Proses *stemming Arifin-Setiono* atau tidak dilakukan.
3. *K-Gram* dan *Modulo* saat proses perhitungan *hashing* pada *Rabin-Karp* bila algoritma yang digunakan adalah pembobotan heuristik *Rabin-Karp.*

3.2.1.2 Perancangan *Prototype* Sistem

Aplikasi yang akan dibuat adalah aplikasi untuk menjawab pertanyaan (*Question Answering*) pada dokumen teks berbahasa Indonesia. Pembuatan aplikasi yang berbasiskan web ini menggunakan bahasa pemrograman php yang terhubung pada *database*. *Database* ini digunakan untuk menyimpan dokumen-dokumen yang akan menjadi jawaban serta tabel-tabel untuk pencarian jawaban yang tepat. Rancangan prototype digunakan sebagai acuan untuk membuat aplikasi dan fitur-fitur yang terdapat pada sistem. Gambar 3.3 adalah gambar rancangan *prototype sistem* dan fitur yang terdapat didalam sistem, antara lain :



Gambar 3.3 Rancangan *prototype system*

Sumber : [Rancangan]

Adapun penjelasan mengenai rancangan user interface diatas adalah sebagai berikut :

1. Textarea yang digunakan untuk input kalimat pertanyaan.
2. *Button proses* untuk menjalankan sistem.
3. *TextField k-gram* untuk menentukan *parsing* sebanyak berapa k-gram.
4. *TextField modulo* untuk menentukan modulo pembagian.

5. *Textarea* hasil *preprocessing* (*case folding*, *tokenizing*, *filtering*, *stemming*).
6. *TextField* hasil *parsing* k-gram.
7. *TextField* hasil *hashing* pada k-gram.
8. *TextField* hasil *count query passages*.
9. *TextField* hasil *count query match*.
10. *TextField* hasil penilaian akhir pembobotan *passages* dengan *heuristic score*.
11. *TextArea* untuk menampilkan hasil proses sistem dalam bentuk kalimat jawaban berdasarkan pertanyaan yang diberikan.
12. *Label* yang menampilkan hasil waktu yang dibutuhkan oleh sistem selama proses berlangsung.

3.2.2 Kebutuhan Data

Data yang dibutuhkan untuk implementasi ini adalah :

1. Data uji berupa dokumen Buku Pedoman Pendidikan Program Teknologi Informasi dan Ilmu Komputer Universitas Brawijaya Tahun Akademik 2012/2013.
2. Data *stoplist / stopwor*d berisi kumpulan kata bahasa Indonesia yang tidak deskriptif atau kurang memiliki makna, berfungsi pada saat proses *filtering*.
3. Data kamus kata dasar yang berisi kumpulan kata dasar bahasa Indonesia yang digunakan dalam proses *stemming*.

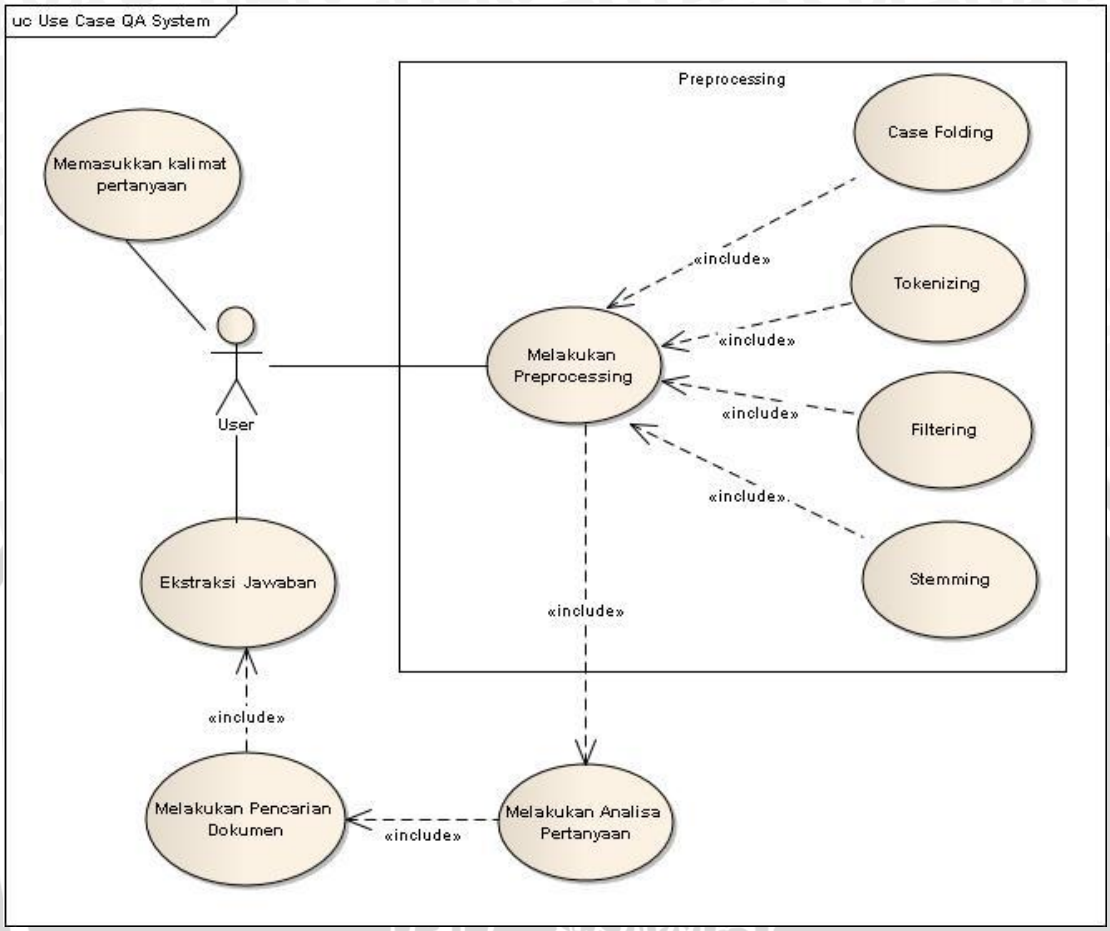
3.2.3 Kebutuhan Fungsional

Fungsi-fungsi yang dimiliki oleh implementasi algoritma dalam sistem adalah sebagai berikut :

1. Program harus mampu melakukan proses *preprocessing*, yakni berupa *case folding*, *tokenizing*, *filtering* penghilangan kata *stopwords* dan *stemming*.
2. Program harus mampu melakukan pembobotan heuristik terhadap kandidat jawaban untuk dilakukan optimasi pencocokan string (*string matching*) menggunakan algoritma *Rabin-Karp.*
3. Program harus mampu melakukan ekstraksi jawaban menggunakan algoritma *Rule-Based.*
4. Program dapat membandingkan perbedaan penggunaan pembobotan heuristik menggunakan algoritma *Rabin-Karp* dan algoritma *Rule-Based* dalam hal kesesuaian jawaban dan waktu proses. Serta membandingkan penggunaan *stemming* atau tidak ketika melakukan proses *preprocessing*.

3.2.4 Diagram UML (*Unified Model Language*)

Untuk mendokumentasikan dan membangun sistem perangkat lunak pada sistem ini digunakan Diagram UML (*Unified Model Language*). UML merupakan teknik untuk pemodelan desain program berorientasi objek (OOP). UML dipilih karena aplikasi yang dibuat menggunakan konsep pemrograman yang berorientasi pada objek atau biasa disebut dengan *Object-oriented programming* (OOP). Untuk merepresentasikannya digunakan *Use Case Diagram*, yang terdiri atas diagram untuk *use case* dan *actor*. *Actor* merepresentasikan orang yang berinteraksi dengan sistem aplikasi, sedangkan *use case* merepresentasikan operasi-operasi yang dilakukan oleh *actor*. Berikut ini adalah rancangan UML *use case diagram* untuk aplikasi *Question Answering System* dapat dilihat pada gambar 3.4



Gambar 3.4 Diagram *Use Case* Sistem

Sumber : [Rancangan]

Setiap *use case* pada gambar 3.4 memiliki scenario *use case* diagram yang akan dijelaskan pada tabel 3.1

Memasukkan kalimat pertanyaan	
Deskripsi Singkat	User memasukkan kalimat pertanyaan pada sistem
Aktor	User
Pre-Condition	Sistem menampilkan form input teks
Post-Condition	Sistem menampilkan kalimat tanya

Melakukan *Preprocessing*	
Deskripsi Singkat	Melakukan tahap *case folding*, *tokenizing*, *filtering* dan *stemming* untuk menata struktural teks
Aktor	User
Relationship	Coba *Case Folding*, Coba *Tokenizing*, Coba *Filtering*, Coba *Stemming*, Melakukan Analisa Pertanyaan
Pre-Condition	Sistem melakukan preprocessing setelah user memasukkan kalimat pertanyaan
Post-Condition	Sistem menghasilkan teks yang lebih struktural dan noise berkurang
Coba *Case Folding*	
Deskripsi Singkat	Mengubah teks menjadi huruf kecil (*lowercase)* semua
Relationship	Melakukan *Preprocessing*
Pre-Condition	Sistem menampilkan form input teks
Post-Condition	Sistem menampilkan form hasil *case folding*
Coba *Tokenizing*	
Deskripsi Singkat	Memecah teks menjadi kumpulan token kata - kata
Relationship	Melakukan *Preprocessing*
Pre-Condition	Sistem menampilkan form input teks
Post-Condition	Sistem menampilkan form hasil *Tokenizing*
Coba *Filtering*	
Deskripsi Singkat	Membuang kata – kata tidak penting yang termasuk dalam *stopwords lists*
Relationship	Melakukan *Preprocessing*
Pre-Condition	Sistem menampilkan form input teks
Post-Condition	Sistem menampilkan form hasil *Filtering*
Coba *Stemming*	
Deskripsi Singkat	Mengembalikan kata - kata ke dalam bentuk kata dasarnya
Relationship	Melakukan *Preprocessing*
Pre-Condition	Sistem menampilkan form input teks
Post-Condition	Sistem menampilkan form hasil *Stemming*

Melakukan Analisa Pertanyaan	
Deskripsi Singkat	Menganalisis kalimat pertanyaan
Relationship	Melakukan *Preprocessing*, Melakukan Pencarian Dokumen
Pre-Condition	Sistem menganalisis kalimat pertanyaan yang telah di-*preprocessing*
Post-Condition	Sistem menghasilkan *keyword* pencarian dan mengidentifikasi tipe dari pertanyaan tersebut sehingga dapat ditentukan *named entity* yang akan dicari untuk dapat menemukan kandidat jawaban dan
Melakukan Pencarian Dokumen	
Deskripsi Singkat	Mencari kandidat jawaban pada *Retrieval Engine*
Relationship	Melakukan Analisa Pertanyaan, Ekstraksi Jawaban
Pre-Condition	Sistem melakukan pencarian pada dokumen yang mengandung informasi yang berkaitan dengan *keyword*
Post-Condition	Sistem menghasilkan kandidat jawaban
Ekstraksi Jawaban	
Deskripsi Singkat	Memberikan informasi akhir dari jawaban yang telah dilakukan penilaian oleh sistem
Aktor	User
Relationship	Melakukan Pencarian Dokumen
Pre-Condition	Sistem melakukan identifikasi terhadap jawaban
Post-Condition	Sistem menampilkan *output* berupa jawaban

Tabel 3.1 Skenario *Use Case* Diagram

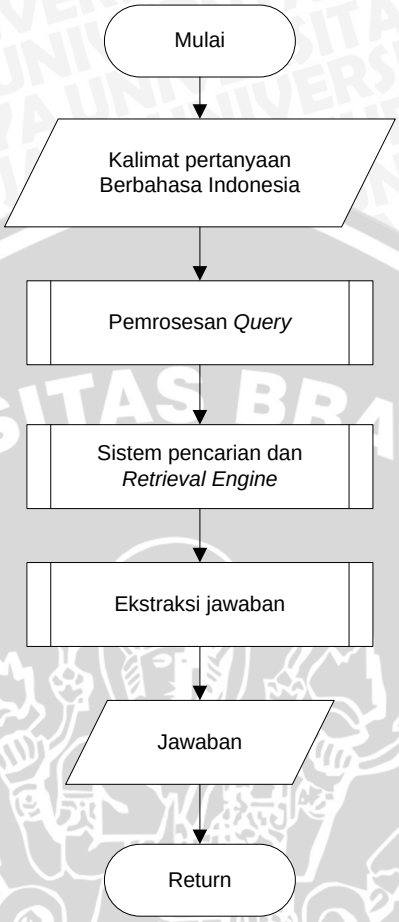
Sumber : [Rancangan]

3.3 Perancangan Sistem

Perancangan sistem dilakukan setelah semua kebutuhan sistem didapatkan melalui tahap analisis sistem. Dengan merancang sistem, nantinya akan memudahkan dalam merunut jalannya proses secara bertahap dalam *Question Answering System*. Hasil dari analisis kebutuhan sistem dapat membantu merancang bagaimana aplikasi berjalan, adapun rancangannya adalah sebagai berikut :

1. Pengguna memasukkan kalimat pertanyaan berbahasa Indonesia.
2. Sistem menganalisis kalimat pertanyaan yang dimasukan secara manual oleh pengguna. Hal ini dimaksudkan untuk mengidentifikasi tipe dari pertanyaan tersebut sehingga dapat ditentukan *named entity* yang akan dicari untuk dapat menemukan kandidat jawaban.
3. Sistem mengubah informasi hasil analisis menjadi *query*. *Query* tanpa kata tanya digunakan untuk memperoleh *n* dokumen teratas.
4. Sistem melakukan pencarian pada dokumen yang mengandung informasi yang berkaitan dengan *query.*
5. Sistem melakukan identifikasi terhadap jawaban. Setiap *n* dokumen teratas yang terambil dianalisis kembali untuk mengidentifikasi kandidat jawaban.
6. Sistem menampilkan output berupa jawaban.

Untuk mempermudah alur jalan dari sistem, maka semua proses yang dijelaskan sebelumnya digambarkan dalam diagram alir sebagai berikut :

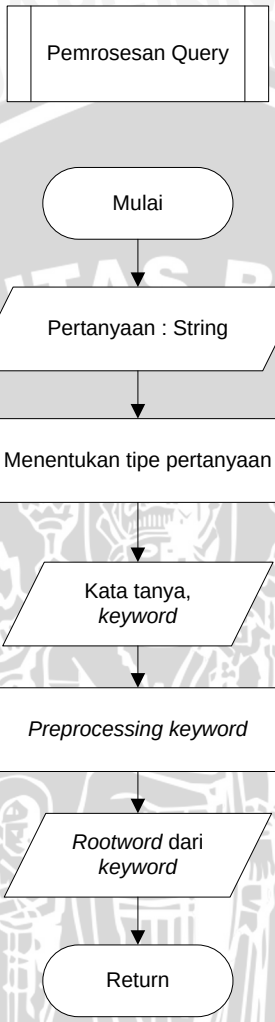


Gambar 3.5 Diagram alir sistem

Sumber : [Rancangan]

Sistem utama dalam *Question Answering System* memiliki tiga modul utama, yakni modul Pemrosesan Query, modul Sistem Pencarian dan Retrieval Engine, serta modul Ekstraksi Jawaban.

3.4 Pemrosesan Query



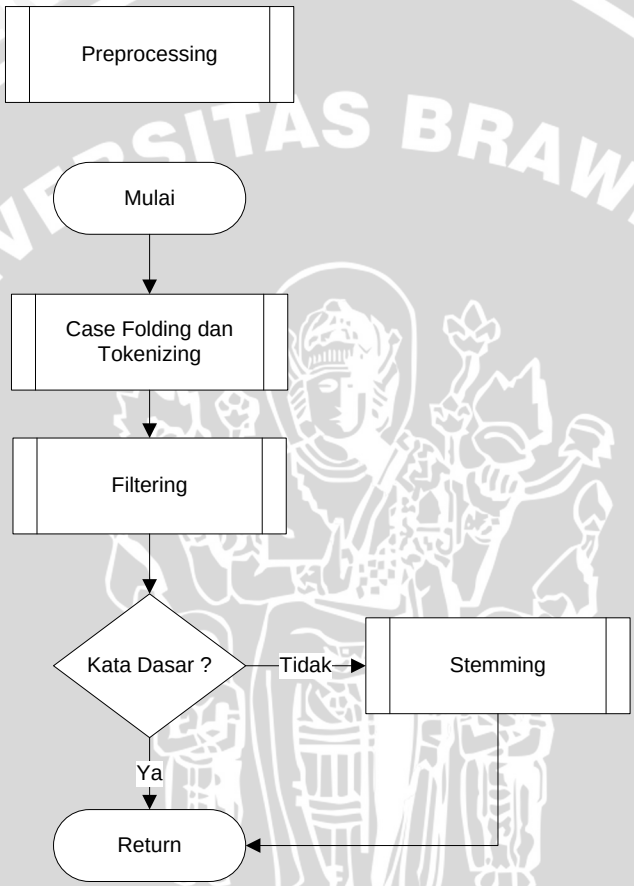
Gambar 3.6 Diagram alir pemrosesan *query*

Sumber : [Rancangan]

Pada pemrosesan *query*, setelah user memasukkan kalimat pertanyaan maka akan ditentukan tipe pertanyaan berdasarkan pola pertanyaan untuk mendapatkan kandidat jawaban berdasarkan *named entity*. Output berupa kata tanya dan *keyword*, kemudian dilakukan *preprocessing terhadap* keyword. Hasil akhir dari pemrosesan

query adalah kata tanya dan *rootword d*ari *keyword* yang disimpan dalam *Retrieval Engine* untuk proses lebih lanjut.

3.4.1 Preprocessing



Gambar 3.7 Diagram alir *preprocessing*

Sumber : [Rancangan]

Tahapan *preprocessing* dilakukan untuk mengurangi *noise* pada data agar lebih terstruktur. Proses-proses yang dilakukan pada *preprocessing* ini adalah *case folding*, *tokenizing*, *filtering*, dan *stemming*.

3.4.1.1 Case Folding dan Tokenizing

Case Folding adalah proses mengubah semua huruf menjadi bentuk huruf kecilnya (*lowercase*). Pada proses *tokenizing* dilakukan penguraian kalimat menjadi kata-kata. Karakter selain huruf dihilangkan dan *dianggap delimiter*. Adapun *delimiter* yang didaftarkan pada sistem ini adalah untuk tanda baca '.', ',', "", """, '-', '/', '{', '}', '+', '_', '!', '@', '#', '$', '%', '^', '&', '*', '(', ')', '?', '>', '<', '[', ']', '|', '`', '~', ';', ':', '=', '\\', "\n", "\r". *Delimiter* juga dianggap sebagai pemisah kata.

Contoh kalimat :

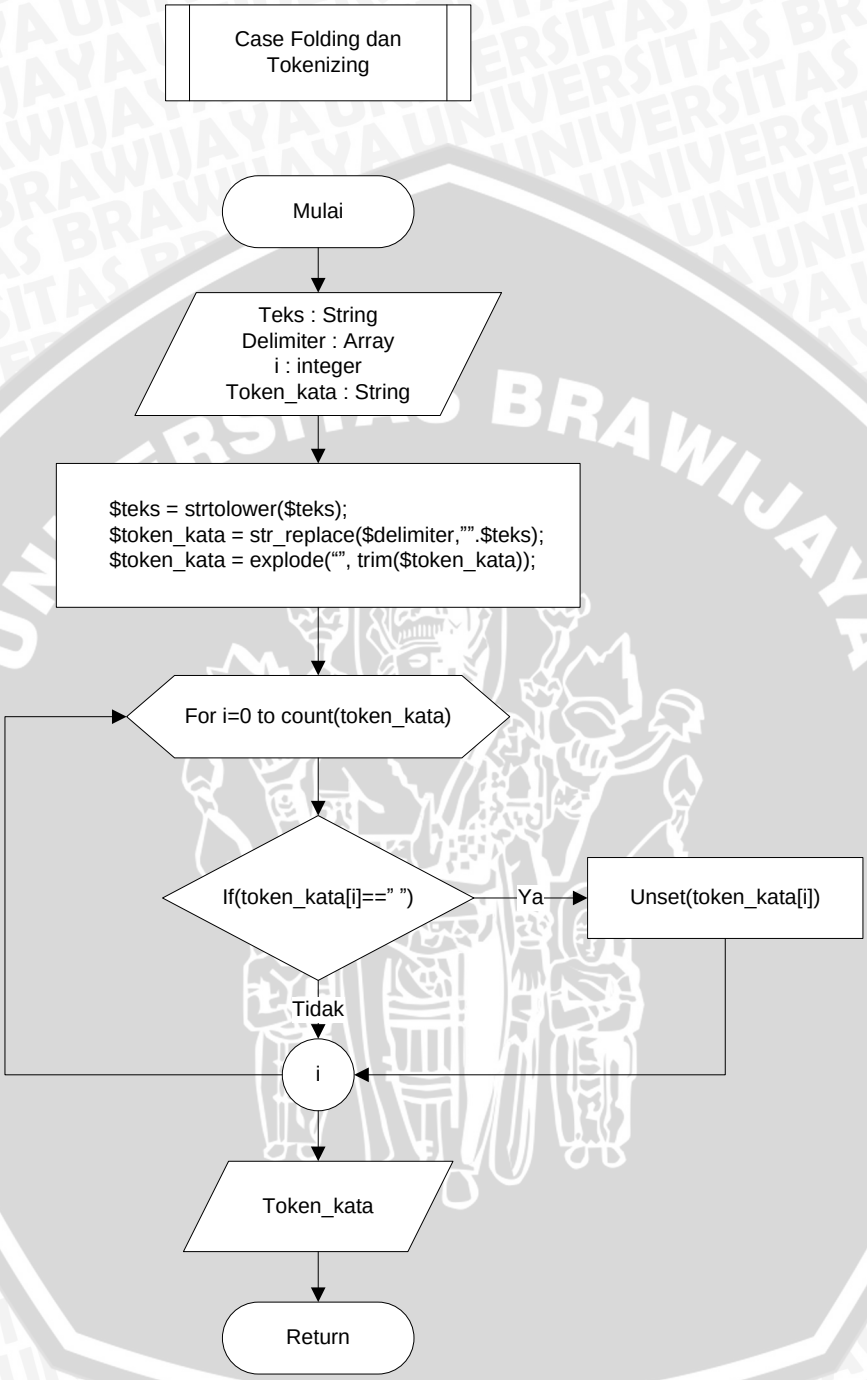
Tugas adalah kegiatan akademik terstruktur yang dilakukan oleh mahasiswa dengan bimbingan dosen/asisten.

Contoh dari *Case Folding* :

tugas adalah kegiatan akademik terstruktur yang dilakukan oleh mahasiswa dengan bimbingan dosen/asisten.

Contoh dari *Tokenizing* :

tugas	adalah	kegiatan	akademik	terstruktur	yang
dilakukan	oleh	mahasiswa	dengan	bimbingan	dosen
asisten					



Gambar 3.8 Diagram alir *case folding* dan *tokenizing*

Sumber : [Rancangan]

3.4.1.2 Filtering

Pada Tahap *filtering* kata - kata penting diambil dari hasil *token*. Metode yang digunakan untuk proses ini adalah metode *stopword* yaitu metode *membuang* kata - kata yang kurang penting. Daftar kata *stopword* disimpan dalam database sebagai kamus, program akan melakukan pengecekan terhadap *stopword*. Kata-kata hasil *tokenizing* dicocokkan dengan tabel *stopword* yang didapatkan dari kamus bahasa Indonesia kemudian disimpan di dalam *database*, bila kata tersebut ditemukan maka kata dihilangkan, tetapi bila kata yang dicari tidak ditemukan dalam database maka kata dijadikan kata penting.

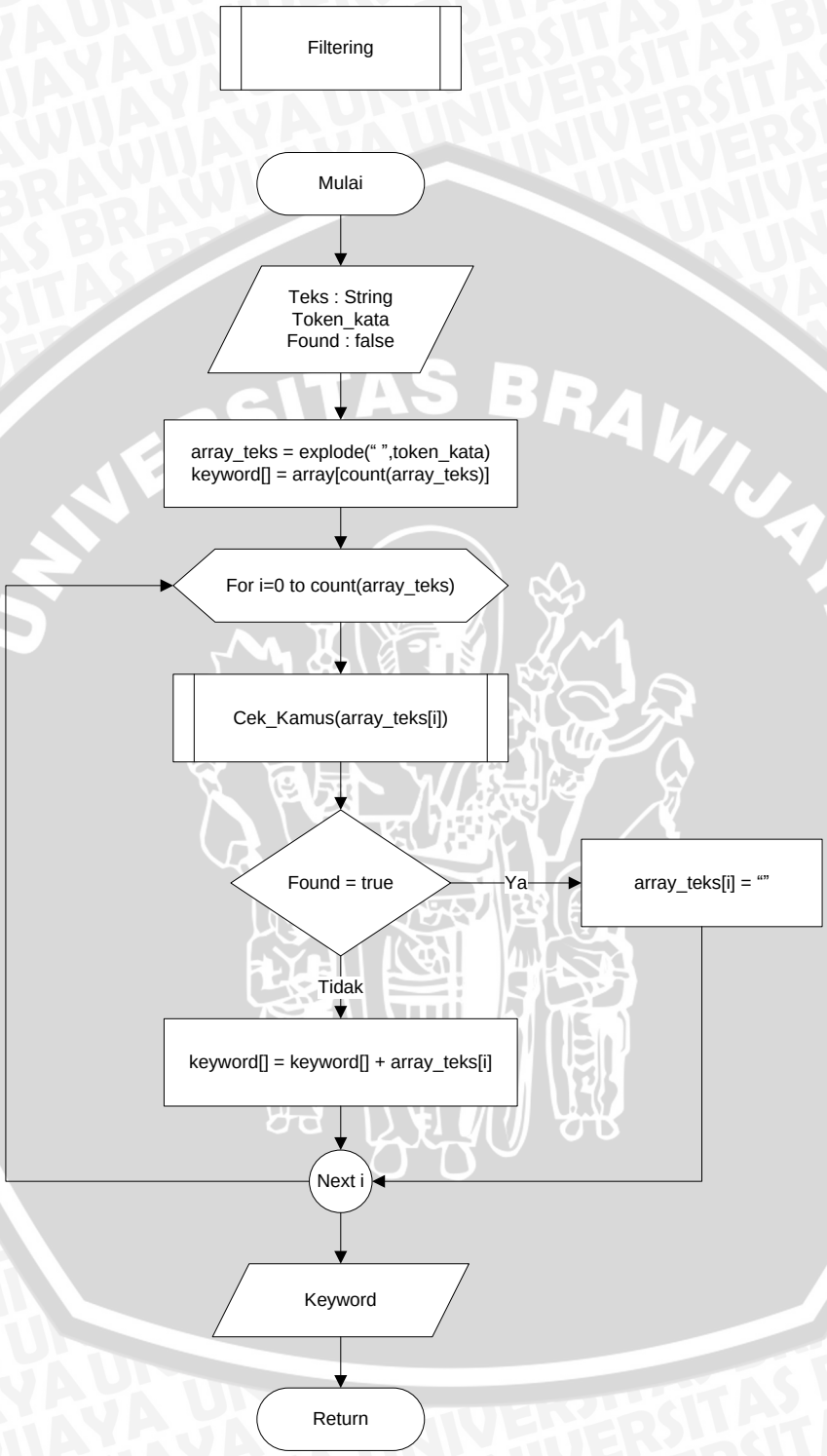
Contoh kalimat :

Tugas adalah kegiatan akademik terstruktur yang dilakukan oleh mahasiswa dengan bimbingan dosen/asisten.

Contoh dari *filtering* :

tugas	kegiatan	akademik	terstruktur	dilakukan	mahasiswa
bimbingan	dosen	asisten			

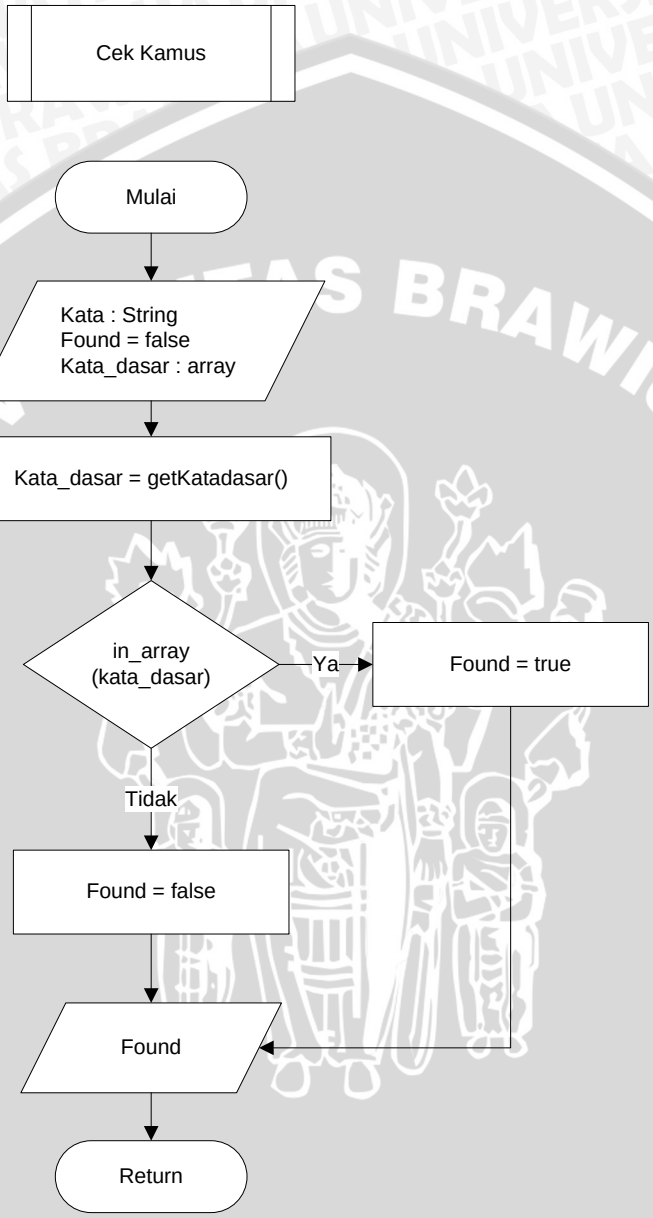
Untuk mempermudah dalam mengerti alur dari filtering, maka proses yang telah dijelaskan sebelumnya digambarkan dalam diagram alir sebagai berikut :



Gambar 3.9 Diagram alir *filtering*

Sumber : [Rancangan]

3.4.1.3 Cek Kamus



Gambar 3.10 Diagram alir cek kamus

Sumber : [Rancangan]

3.4.1.4 Stemming

Stemming adalah proses mentransformasi kata-kata dalam dokumen teks ke bentuk kata dasarnya dengan menghilangkan semua imbuhan (*affixes)* baik yang terdiri dari awalan (*prefixes*), akhiran (*suffixes*) dan *confixes* (kombinasi dari awalan dan akhiran). Sebagai contoh kata berjalan, menjalankan, perjalanan, akan distem ke *root word*-nya yaitu "jalan". Pertanyaan yang sudah dinormalisasi inilah yang akan menjadi kata kunci (*keyword*) dalam pencarian dokumen. Algoritma *stemming* yang digunakan dalam sistem ini adalah *Arifin-Setiono.*

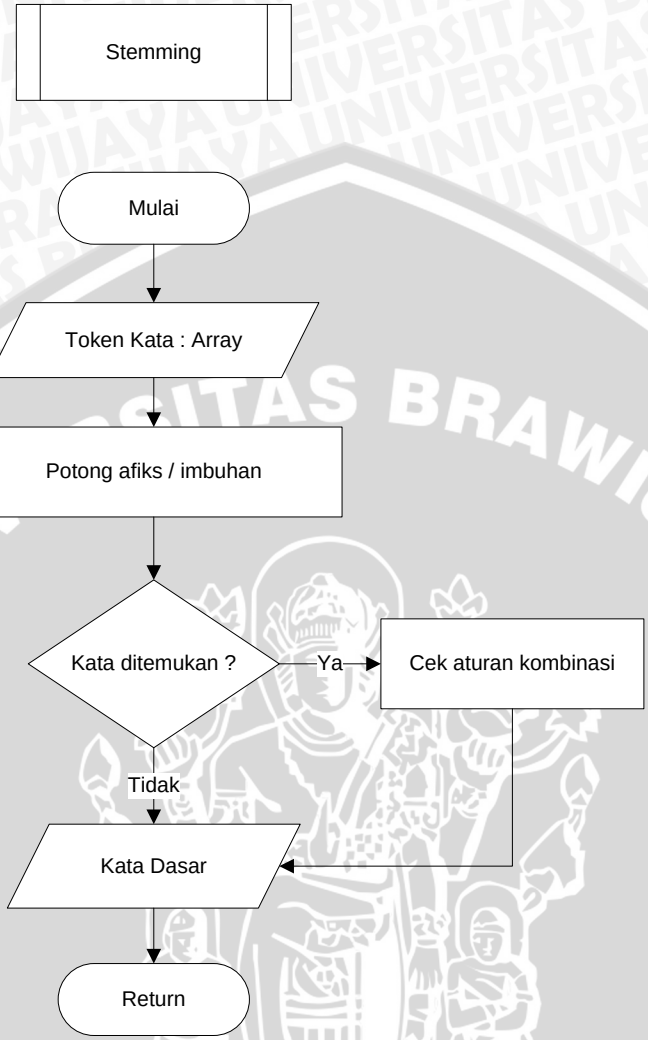
Contoh kalimat :

Tugas adalah kegiatan akademik terstruktur yang dilakukan oleh mahasiswa dengan bimbingan dosen/asisten.

Contoh dari *stemming* :

tugas	giat	akademik	struktur	laku	mahasiswa	bimbing
dosen	asisten					

Algoritma stemming *Arifin-Setiono* secara lengkap dapat dilihat pada lampiran. Untuk mempermudah dalam mengerti alur dari stemming, maka proses yang telah dijelaskan sebelumnya digambarkan dalam diagram alir sebagai berikut :



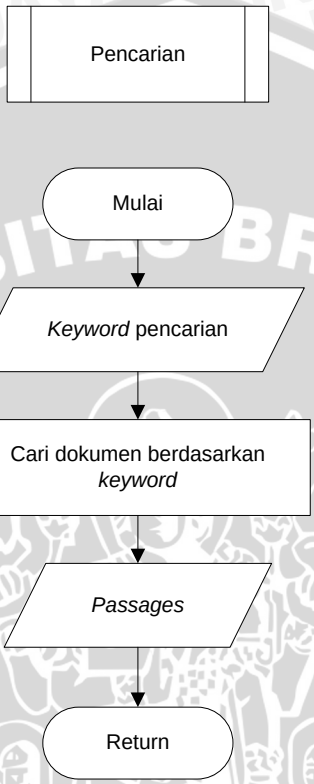
Gambar 3.11 Diagram alir *stemming*

Sumber : [Rancangan]

3.5 Pencarian dan Retrieval Engine

Search Engine mencari dokumen-dokumen yang relevan dengan jawaban yang diinginkan dari pertanyaan dengan menggunakan query yang dihasilkan Pemrosesan *Query*. Dokumen-dokumen yang diperoleh akan menjadi kandidat

jawaban dan dokumen-dokumen ini masih perlu melalui beberapa proses sebelum didapatkan jawaban yang diinginkan.

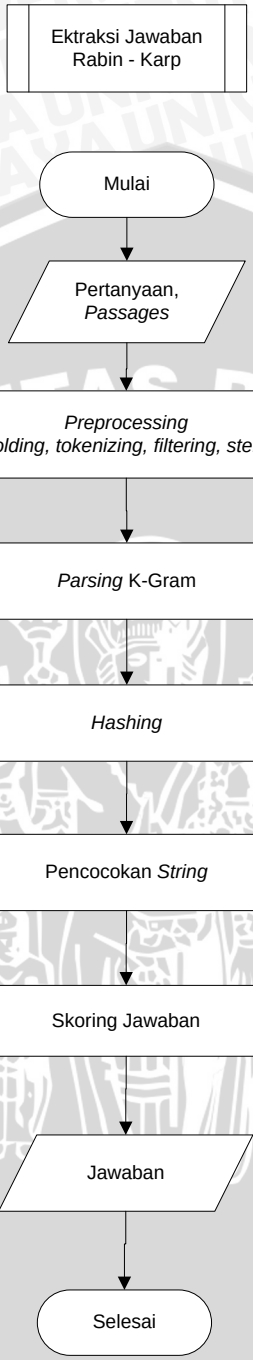


Gambar 3.12 Diagram alir pencarian

Sumber : [Rancangan]

3.6 Ekstraksi Jawaban dengan Pembobotan Rabin-Karp

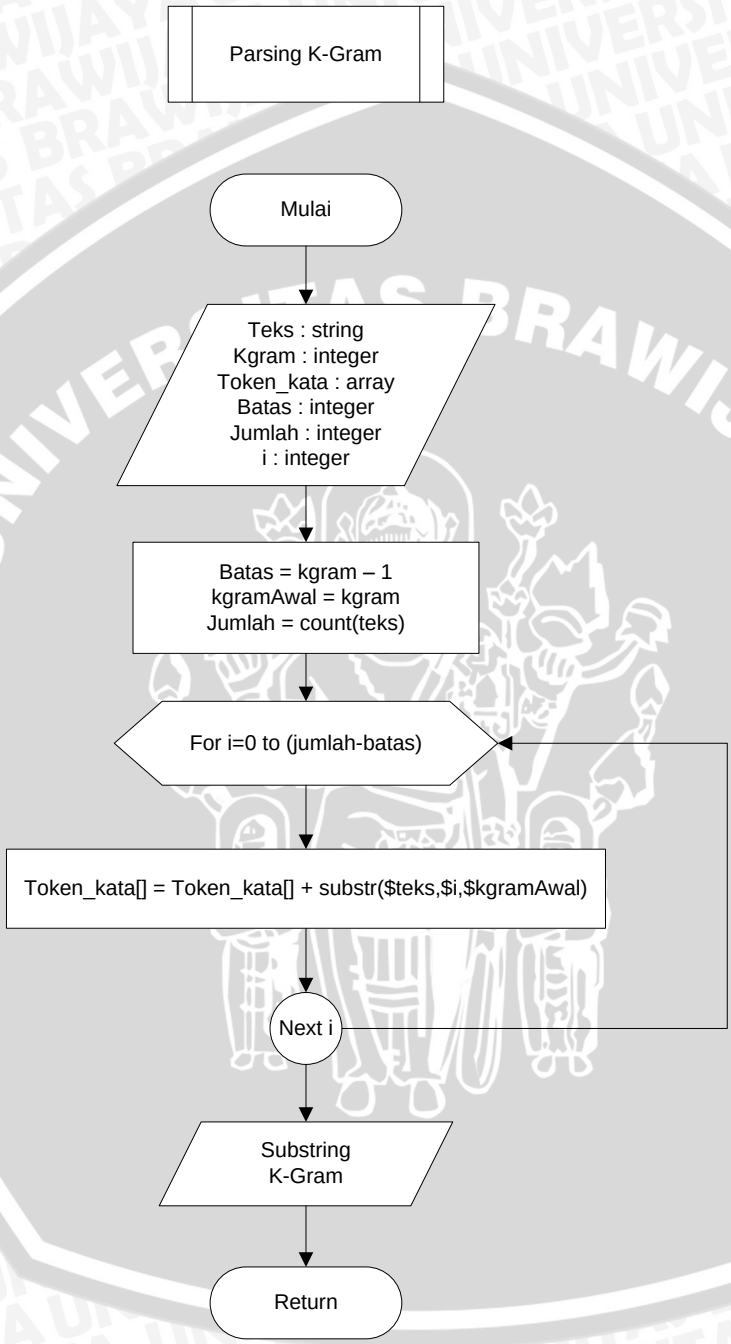
Setelah *passages* dibentuk pada proses pencarian, jawaban dihitung dengan menggunakan pembobotan heuristik menggunakan algoritma *Rabin-Karp*. *Passages* diberikan pembobotan dengan *heuristic score*, jarak terdekat dengan *query* akan dipilih sebagai jawaban. Berikut ini adalah diagram alir ekstraksi jawaban dengan pembobotan heuristik *Rabin-Karp* :



Gambar 3.13 Diagram alir pembobotan Rabin-Karp

Sumber : [Rancangan]

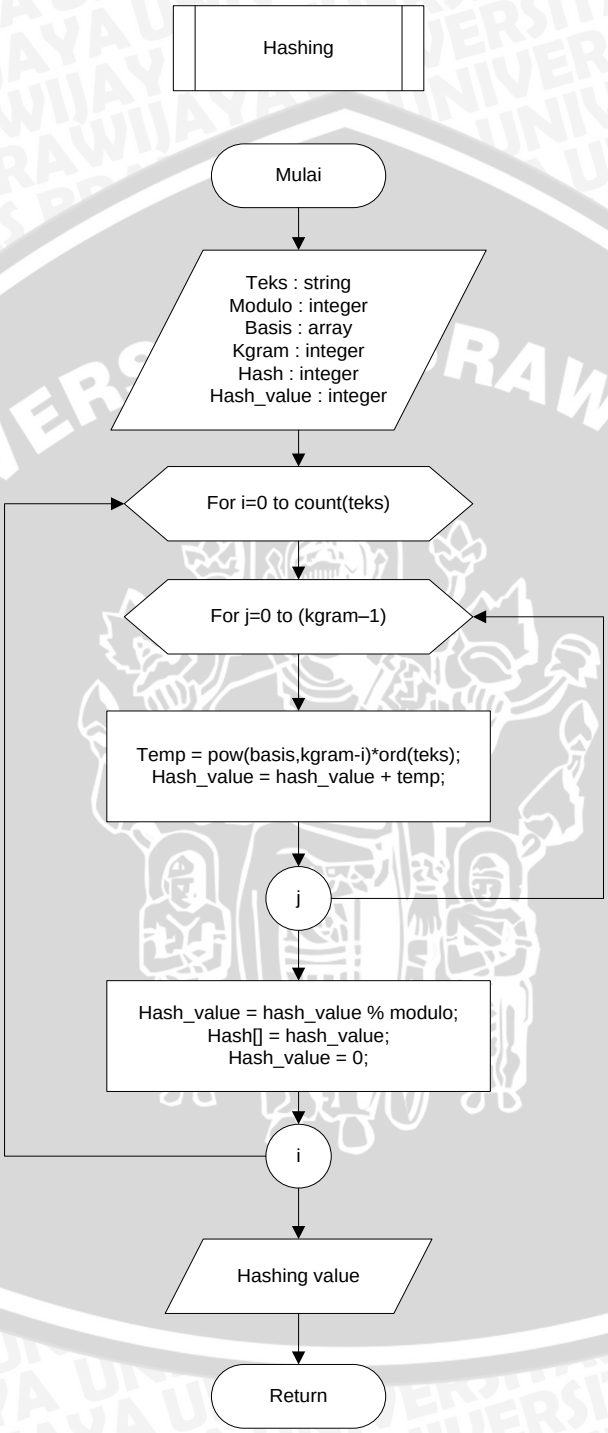
3.6.1 Parsing K-Gram



Gambar 3.14 Diagram alir *parsing* K-Gram

Sumber : [Rancangan]

3.6.2 Hashing

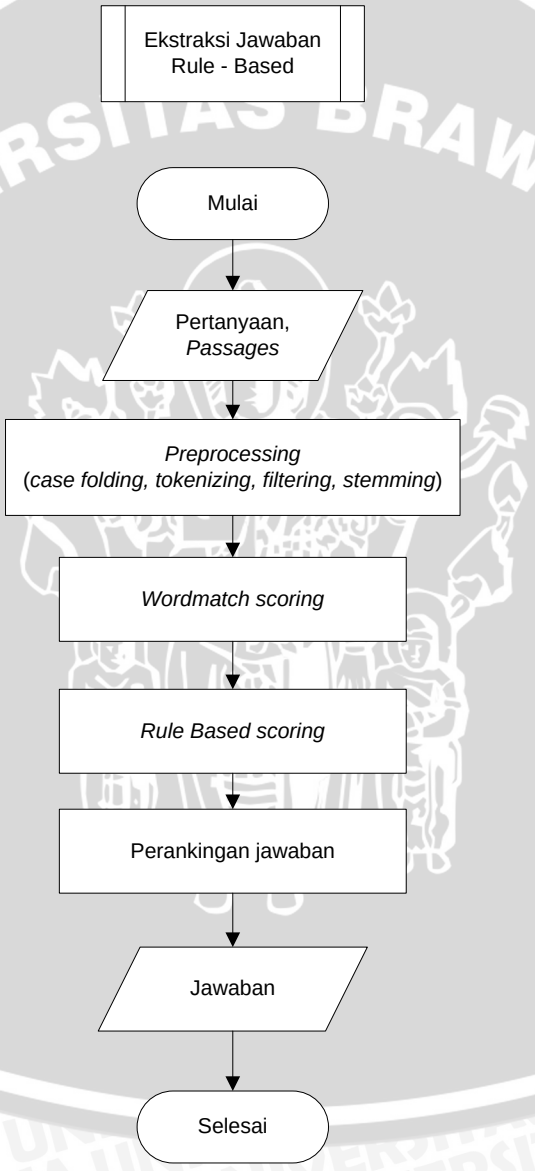


Gambar 3.15 Diagram alir *hashing*

Sumber : [Rancangan]

3.7 Ekstraksi Jawaban dengan Pembobotan Rule-Based

Pencarian jawaban akhir juga dilakukan dengan scoring jawaban dengan Pembobotan Rule-Based sesuai penelitian yang telah dilakukan oleh Ikhsani (2006). Hal ini dilakukan sebagai pembanding untuk dilakukan pengujian.



Gambar 3.16 Diagram alir pembobotan Rule-Based

Sumber : [Rancangan]

3.8 Implementasi

Implementasi dilakukan dengan mengacu kepada perancangan aplikasi. Yang terdiri dari tiga komponen utama, yaitu : Pemrosesan *Query*, Modul Sistem Pencarian, dan Ekstraksi Jawaban.

3.8.1 Pengumpulan Data

Pengumpulan data dilakukan untuk memperoleh data-data yang dibutuhkan oleh sistem, yang meliputi dokumen, pola pertanyaan, dan pola jawaban. Bagian ini menjelaskan bagaimana data didapatkan dan disimpan ke dalam *database*.

Adapun dokumen yang digunakan dalam sistem ini berasal dari Buku Pedoman Pendidikan. Data yang diperoleh dalam bentuk *softcopy* dengan format *pdf*. Dokumen - dokumen ini dimasukkan ke dalam *database* dengan menggunakan beberapa *tools*. Penggunaan *tools* dipilih supaya proses memasukkan data ke dalam database dapat dilakukan dengan lebih cepat.

Tools yang digunakan untuk memasukkan data berupa aplikasi web dalam bentuk php dimana pengguna mengisikan field-field pada form yang nantinya akan disimpan dalam database. Pengguna juga harus memasukkan kategori named entity tag yang sesuai dengan *DESCRIPTION*, *NAME*, *LOCATION*, *ORGANIZATION*, *DATE, TIME*, *CURRENCY* dan *NUMBER*.

Proses memasukkan data secara manual juga dilakukan karena beberapa dokumen tidak memungkinkan untuk dimasukkan secara otomatis ataupun dengan menggunakan *tools*. Dokumen-dokumen tersebut meliputi dokumen yang berupa gambar, tabel, maupun diagram alir, dan dokumen-dokumen lain yang bukan berupa teks. Dalam memasukkan dokumen-dokumen tersebut, pertama dokumen diubah ke dalam bentuk teks secara manual dan kemudian dimasukkan ke dalam database.

Ketika semua dokumen telah dimasukkan ke dalam database, proses selanjutnya adalah dengan melakukan *preprocessing* pada dokumen tersebut.

3.8.1 Pemrosesan *Query*

Sistem menerima input berupa kalimat pertanyaan. Selanjutnya dilakukan normalisasi pada string pertanyaan yang dimasukkan, seperti dengan menghilangkan tanda baca, dan mengganti huruf kapital dengan huruf kecil.

Pertanyaan yang sudah dinormalisasi kemudian dianalisis dengan menginterpretasikannya. Pertanyaan diinterpretasikan dengan memanfaatkan pola pertanyaan (*Question Patterns*) yang sudah dipersiapkan. Sebagai langkah awal dilakukan pendefinisian kembali tipe pertanyaan yang memiliki hubungan dengan NE (*named entity*) yang baku. Mengacu pada penelitian Ballesteros & Xiaoyan-Li (2007), ada tujuh tipe pertanyaan yang terdiri atas *DESCRIPTION*, *NAME*, *LOCATION*, *ORGANIZATION*, *DATE*, *TIME*, *CURRENCY* dan *NUMBER*. Dalam bahasa Indonesia akan diwakili dengan APA, SIAPA, DIMANA, KAPAN, dan BERAPA. Setiap pertanyaan akan mengalami proses *parsing* terlebih dahulu.

Tabel 3.2 Daftar pasangan kata tanya dan *named entity*

No	**Kata Tanya**	***Name Entity Tag***
1	Apa	*DESCRIPTION*
2	Siapa	*NAME, ORGANIZATION*
3	Dimana	*LOCATION*
4	Kapan	*DATE,TIME*
5	Berapa	*NUMBER, PERCENT, CURRENCY*

3.8.2 Modul Sistem Pencarian

Hasil dari Pemrosesan *Query* digunakan oleh sistem pencarian untuk diproses dalam *Retrieval Engine*. Hal ini perlu untuk melakukan pemeringkatan dokumen yang mengandung informasi yang berkaitan dengan *query*.

3.8.3 Modul Ekstraksi Jawaban

Pada tahap ini dilakukan identifikasi terhadap jawaban. Setiap *n* dokumen teratas yang terambil dianalisis kembali untuk mengidentifikasi kandidat jawaban.

a) Perolehan Top Dokumen

Proses ini dilakukan untuk mendapatkan dokumen dengan skor tertinggi yang akan digunakan untuk membentuk kalimat dan *passage* pada tahap selanjutnya. *Query* yang sudah diformat akan dijalankan terhadap indeks yang telah dibuat, kemudian dilakukan pembobotan dokumen.

b) Pembentukan Passages

Dokumen yang telah ditemukembalikan kemudian diproses menjadi kalimat. *Passages* yang diperoleh kemudian di-*filter berdasarkan tag name entity* yang sesuai dengan kata tanya. Misalnya kata tanya "Siapa" maka *passages* yang diperoleh adalah yang mengandung tag ORGANIZATION atau PERSON. Kemudian *passages* yang telah diseleksi disimpan untuk proses selanjutnya.

c) Pembobotan dengan Rabin - Karp

Passages yang telah diseleksi kemudian diberi bobot atau skor dengan Rabin-Karp. Proses penghitungan skor tiap *passage* dilakukan dengan menjumlahkan semua kemungkinan kata yang cocok dengan kueri

pertanyaan. Pembobotan dilakukan dengan pembobotan heuristik, Proses pembobotannya adalah sebagai berikut :

1. Jika tidak ada *named entity* yang ditampilkan, *passage* menerima nilai 0. Jika *named entity* ditampilkan pada *passage* namun tidak memiliki tipe yang sama dengan pertanyaan, *named entity* diabaikan.
2. Dilakukan pencocokan kata-kata pada query dengan kata-kata pada *passage* (proses *wordmatch*). Jika nilai *count_match* kurang dari *threshold* (t), *score* = 0. Selain itu, *score = count_match*. Nilai *threshold* (t), didefinisikan dengan cara berikut :
    a. Jika count_queri kurang dari 4, t=count_queri.
    b. Jika count_queri antara 4 dan 8, t=count_queri/2.0+1.0.
    c. Jika lebih besar dari 8, t=count_queri/3.0+2.0.

    Nilai *threshold* digunakan untuk mengambil kata yang penting pada *passages*. Dengan kata lain, paragraf apapun yang tidak mengandung kata-kata yang terdapat pada query tidak diperhitungkan.
3. Kata yang berdekatan memiliki hubungan keterkaitan informasi yang lebih tinggi. Jika seluruh kata yang cocok dengan *query* terdapat pada satu *passages* Sm=1, selain itu Sm=0. Dengan demikian, *score = score* + (Sm*0.5).
4. Seperti yang diketahui urutan kata dapat mempengaruhi arti. Oleh karena itu, diberikan bobot yang lebih tinggi (Ord=1) terhadap *passage* jika kata-kata yang cocok dengan *query* memiliki urutan yang sama seperti pada pertanyaan asli. Selain itu Ord=0. Dengan demikian, *score = score* + (Ord*0.5).
5. *Score = score* + (*count_match*/W), dengan W adalah jumlah kata dari *passage* dengan bobot tertinggi.

Pembobotan terakhir yaitu menghitung total perolehan nilai yang disimpan dalam variabel *heuristic_score* yaitu *count_match* + 0.5*Sm + 0.5*Ord + *count_match*/W

d) Pembobotan dengan Rule - Based

Tahap rule based scoring merupakan tahap pemberian skor bedasarkan kata tanya. Setiap kata tanya memiliki rule atau aturan tertentu. Skor pada masing-masing rule berbeda satu dan yang lainnya.

Skor bervariasi yaitu +3, +4, +6, dan +20. Pemberian skor ini berdasarkan perkiraan seberapa penting sebuah rule. Pertanyaan dengan kata tanya SIAPA merupakan pertanyaan yang menghendaki jawaban subjek atau orang tertentu. Sehingga jawaban dari pertanyaan siapa bisa berupa nama orang, kata ganti orang, atau pekerjaan dari seseorang. Aturan untuk pertanyaan SIAPA adalah sebagai berikut:

```
Score(S) += WordMatch(Q,S)
If ¬ contains (Q, NAMA) and contains (S, "nama") Then
      Score(S) += 4
If contains (S, {NAMA, ORANG})
      Then Score(S) += 4
```

Pertanyaan DIMANA merupakan pertanyaan yang menghendaki jawaban yang berupa keterangan tempat. Aturan pertanyaan DIMANA adalah sebagai berikut:

```
Score(S) += WordMatch(Q,S)
If contains (S, preposisi tempat) ThenScore += 4
If contains (S, keterangan tempat) ThenScore += 6
```

Pertanyaan dengan kata tanya KAPAN merupakan pertanyaan yang menanyakan waktu terjadinya peristiwa, sehingga pertanyaan mengapa mencari jawaban yang di dalamnya terdapat kata "mulai", "sejak", "awal" atau "akhir". Aturan untuk pertanyaan mengapa adalah sebagai berikut:

```
Score(S) += WordMatch(Q,S)
If contains (S, WAKTU) Then Score += 4
If contains (S, penunjuk durasi waktu) Then Score += 4
If contains (Q, "akhir") and contains
(S,{"sejak","akhir"})ThenScore += 20
If contains (Q, {"mulai", "awal"}) and contains
(S,{"mulai", "sejak" })ThenScore += 20
```

Pertanyaan APA merupakan pertanyaan yang menanyakan benda. Aturan untuk pertanyaan yang menggunakan kata tanya APA adalah sebagai berikut:

```
Score(S) += WordMatch(Q,S)
If contains (S, {"ialah", "adalah"}) Then Score += 4
If contains (Q, {"maksud"}) and contains (S,
     {"adalah","ialah","yaitu"}) Then Score += 20
```

e) **Perolehan Kandidat Jawaban**

Passage dengan bobot tertinggi akan dipilih sebagai *Top Passage*. Langkah selanjutnya adalah mencari kata-kata dalam tag yang sesuai dengan kata tanya. Semua daftar kandidat jawaban yang sesuai disimpan dalam suatu *array* untuk diproses pada tahap pengambilan entitas jawaban.

3.9 Perhitungan manual

1. Pembobotan Rabin-Karp

Pertanyaan :

Apa itu semester pendek ?

Preprocessing pertanyaan :

Kata tanya : APA

Keyword : | semester | pendek |

Contoh dokumen sebelum preprocessing :

1. Semester adalah satuan waktu kegiatan yang terdiri atas 16 minggu kuliah atau kegiatan terjadwal lainnya, berikut kegiatan iringannya, termasuk 2 sampai 3 minggu kegiatan penilaian.
2. Semester pendek adalah suatu semester untuk mahasiswa tertentu yang penyelenggaraannya diperpendek dengan mengacu pada ketentuan yang ditetapkan oleh Ketua Progam.
3. Ujian Akhir Semester adalah kegiatan evaluasi yang dilakukan pada akhir semester.
4. Program semester pendek bertujuan memberikan kesempatan kepada mahasiswa untuk memperbaiki nilai mata kuliah yang sudah pernah ditempuh dalam rangka meningkatkan indeks prestasi kumulatif dan memperpendek masa studi serta menghindari terjadinya putus studi.
5. Penyelenggaraan program semester pendek meliputi kegiatan tatap muka, praktikum (bagi mata kuliah berpraktikum), tugas terstruktur, tugas mandiri, ujian tengah dan ujian akhir.
6. Kurikulum dan peraturan akademik dalam perkuliahan semester pendek tetap mengacu pada kurikulum dan peraturan akademik yang berlaku saat itu.
7. Semester pendek tidak diperhitungkan dalam perhitungan masa studi.

Hasil akhir pencarian dokumen setelah preprocessing berdasarkan keyword :

1. semester satu waktu giat atas 16 minggu kuliah giat jadwal ikut giat iringannya masuk 2 3 minggu giat penilaian

2. semester pendek semester untuk mahasiswa selenggara pendek kacu tentu yang tetap oleh ketua progam
3. uji akhir semester giat evaluasi laku akhir semester
4. program semester pendek tuju ikan sempat mahasiswa untuk baik nilai mata kuliah tempuh rangka meningkatkan indeks prestasi kumulatif pendek masa studi serta hindar jadi putus studi
5. selenggara program semester pendek liput giat tatap muka praktikum mata kuliah praktikum tugas struktur tugas mandiri uji tengah uji akhir
6. kurikulum atur akademik kuliah semester pendek tetap kacu kurikulum atur akademik laku saat itu
7. semester pendek hitung hitung masa studi

Dilakukan identifikasi named entity yang terdiri atas orang, organisasi, lokasi, ekspresi waktu, tanggal, ekspresi numerik dan uang. Untuk APA kandidat berdasarkan keyword :

1. Semester adalah satuan waktu kegiatan yang terdiri atas 16 minggu kuliah atau kegiatan terjadwal lainnya, berikut kegiatan iringannya, termasuk 2 sampai 3 minggu kegiatan penilaian.
2. Semester pendek adalah suatu semester untuk mahasiswa tertentu yang penyelenggaraannya diperpendek dengan mengacu pada ketentuan yang ditetapkan oleh Ketua Program.
3. Ujian Akhir Semester adalah kegiatan evaluasi yang dilakukan pada akhir semester.

Preprocessing kandidat jawaban :

1. semester satu waktu giat atas 16 minggu kuliah giat jadwal ikut giat iringannya masuk 2 3 minggu giat penilaian
2. semester pendek semester untuk mahasiswa selenggara pendek kacu tentu yang tetap oleh ketua program
3. uji akhir semester giat evaluasi laku akhir semester

Ekstraksi jawaban :

Untuk Kandidat 1

Query **:** semester pendek

Kandidat **:** semester satu waktu giat atas 16 minggu kuliah giat jadwal ikut giat iringannya masuk 2 3 minggu giat penilaian

parsing k-gram query

| seme | emes | mest | este | ster | terp | erpe | rpen | pend | ende | ndek |

parsing k-gram kandidat

| seme | emes | mest | este | ster | ters | ersa | rsat | satu | atuw | tuwa | uwak | wakt | aktu | ktug | tugi | ugia | giat | iata | atat | tata | atas | tas1 | as16 | s16m | 16mi | 6min | ming | ingg | nggu | gguk | guku | ukul | kuli | ulia | liah | iahg | ahgi | hgia | giat | iatj | atja | tjad | jadw | adwa | dwal | wali | alik | liku | ikut | kutg | utgi | tgia | giat | iati | atir | tiri | irin | ring | inga | ngan | gann | anny | nnya | nyam | yama | amas | masu | asuk | suk2 | uk23 | k23m | 23mi | 3min | ming | ingg | nggu | ggug | gugi | ugia | giat | iatp | atpe | tpen | peni | enil | nila | ilai | laia | aian |

Setelah melakukan parsing k-gram, langkah selanjutnya adalah melakukan parsing hashing terhadap substring k-gram hasil parsing k-gram. Berikut adalah sedikit contoh perhitungan dari proses hashing :

k-gram = 4 ; basis = 10 ; modulo = 101 ;

pattern **= seme**

hash = $(115*10^3 + 101*10^2 + 109*10^1 + 101*10^0) \bmod 101$

= $(115000 + 10100 + 1090 + 101) \bmod 101$

= $126291 \bmod 101 = 41$

pattern **= emes**

hash = $(101*10^3 + 109*10^2 + 101*10^1 + 115*10^0) \bmod 101$

= (101000 + 10900 + 1010 + 115) mod 101

= 113025 mod 101 = 6

dst. Sehingga didapatkan :

hashing query :

| 41 | 6 | 75 | 35 | 60 | 92 | 97 | 70 | 80 | 82 | 18 |

hashing kandidat :

| 41 | 6 | 75 | 35 | 60 | 95 | 22 | 33 | 30 | 1 | 10 | 91 | 0 | 99 | 87 | 60 | 76 | 52 | 9 | 0 | 0 | 100 | 43 | 65 | 56 | 45 | 6 | 8 | 74 | 45 | 43 | 40 | 1 | 99 | 71 | 91 | 97 | 61 | 4 | 52 | 18 | 71 | 6 | 63 | 15 | 60 | 100 | 79 | 2 | 28 | 76 | 51 | 86 | 52 | 17 | 78 | 81 | 97 | 59 | 68 | 79 | 83 | 40 | 97 | 60 | 82 | 6 | 80 | 91 | 55 | 82 | 4 | 38 | 36 | 8 | 74 | 45 | 39 | 89 | 76 | 52 | 24 | 34 | 50 | 85 | 38 | 73 | 18 | 71 | 5 |

Skoring passages :

count query = 11, count match = 5

count query > 8, maka treshold(t) = (count_query / 3) + 2 = (11/3) + 2 = 5.67

count match < treshold, maka score = 0

score = score + Sm*0.5 = 0

score = score + Ord*0.5 = 0

score = score + count_match/W = 5 / 90 = 0.05

Heuristic_score = count_match + 0.5*Sm + 0.5*Ord + count_match/W

= 0 + 0 +0 + 0.05 = 0.05

Untuk Kandidat 2

Query : semester pendek

Kandidat **:** semester pendek semester untuk mahasiswa selenggara pendek kacu tentu yang tetap oleh ketua program

parsing k-gram query

| seme | emes | mest | este | ster | terp | erpe | rpen | pend | ende | ndek |

parsing k-gram kandidat

| seme | emes | mest | este | ster | terp | erpe | rpen | pend | ende | ndek | deks | ekse | ksem | seme | emes | mest | este | ster | teru | erun | runt | untu | ntuk | tukm | ukma | kmah | maha | ahas | hasi | asis | sisw | iswa | swas | wase | asel | sele | elen | leng | engg | ngga | ggar | gara | arap | rape | apen | pend | ende | ndek | dekk | ekka | kkac | kacu | acut | cute | utet | teta | etap | tapk | apke | pket | ketu | etua | tuap | uapr | apro | prog | rogr | ogra | gram |

Setelah melakukan parsing k-gram, langkah selanjutnya adalah melakukan parsing hashing terhadap substring k-gram hasil parsing k-gram. Berikut adalah sedikit contoh perhitungan dari proses hashing :

k-gram = 4 ; basis = 10 ; modulo = 101 ;

pattern **= seme**

hash = $(115*10^3 + 101*10^2 + 109*10^1 + 101*10^0) \bmod 101$

= $(115000 + 10100 + 1090 + 101) \bmod 101$

= $126291 \bmod 101 = 41$

pattern **= emes**

hash = $(101*10^3 + 109*10^2 + 101*10^1 + 115*10^0) \bmod 101$

= $(101000 + 10900 + 1010 + 115) \bmod 101$

= $113025 \bmod 101 = 6$

dst. Sehingga didapatkan :

hashing query :

| 41 | 6 | 75 | 35 | 60 | 92 | 97 | 70 | 80 | 82 | 18 |

hashing kandidat :

| 41 | 6 | 75 | 35 | 60 | 92 | 97 | 70 | 80 | 82 | 18 | 84 | 33 | 35 | 41 | 6 | 75 | 35 | 60 | 97 | 55 | 60 | 98 | 61 | 3 | 11 | 97 | 51 | 11 | 17 | 80 | 14 | 21 | 18 | 65 | 33 | 31 | 2 | 22 | 13 | 25 | 52 | 9 | 99 | 85 | 38 | 80 | 82 | 18 | 76 | 50 | 94 | 41 | 15 | 53 | 42 | 97 | 57 | 71 | 89 | 0 | 5 | 40 | 7 | 68 | 68 | 80 | 95 | 24 | 36 |

Skoring passages :

count query = 11, count match = 11

count query > 8, maka treshold(t) = (count_query / 3) + 2 = (11/3) + 2 = 5.67

count match >= treshold, maka score = count match = 11

score = score + Sm*0.5 = 0.5

score = score + Ord*0.5 = 0.5

score = score + count_match/W = 11 / 70 = 0.16

Heuristic_score = count_match + 0.5*Sm + 0.5*Ord + count_match/W

= 11 + 0.5 + 0.5 + 0.16 = 12.16

Untuk Kandidat 3

Query : semester pendek

Kandidat : uji akhir semester giat evaluasi laku akhir semester

parsing k-gram query

| seme | emes | mest | este | ster | terp | erpe | rpen | pend | ende | ndek |

parsing k-gram kandidat

| ujia | jiak | iakh | akhi | khir | hirs | irse | rsem | seme | emes | mest | este | ster | terg | ergi | rgia | giat | iate | atev | teva | eval | valu | alua | luas | uasi | asil | sila | ilak | laku | akua | kuak | uakh | akhi | khir | hirs | irse | rsem | seme | emes | mest | este | ster |

Setelah melakukan parsing k-gram, langkah selanjutnya adalah melakukan parsing hashing terhadap substring k-gram hasil parsing k-gram. Berikut adalah sedikit contoh perhitungan dari proses hashing :

k-gram = 4 ; basis = 10 ; modulo = 101 ;

pattern = seme

hash = $(115*10^3 + 101*10^2 + 109*10^1 + 101*10^0) \bmod 101$

= $(115000 + 10100 + 1090 + 101) \bmod 101$

= $126291 \bmod 101 = 41$

pattern = emes

hash = $(101*10^3 + 109*10^2 + 101*10^1 + 115*10^0) \bmod 101$

= $(101000 + 10900 + 1010 + 115) \bmod 101$

= $113025 \bmod 101 = 6$

dst. Sehingga didapatkan :

hashing query :

| 41 | 6 | 75 | 35 | 60 | 92 | 97 | 70 | 80 | 82 | 18 |

hashing kandidat :

| 73 | 13 | 27 | 68 | 91 | 9 | 87 | 66 | 41 | 6 | 75 | 35 | 60 | 83 | 11 | 5 | 52 | 13 | 42 | 16 | 51 | 21 | 88 | 90 | 89 | 73 | 23 | 20 | 10 | 89 | 92 | 8 | 68 | 91 | 9 | 87 | 66 | 41 | 6 | 75 | 35 | 60 |

Skoring passages :

count query = 11, count match = 5

count query > 8, maka treshold(t) = (count_query / 3) + 2 = (11/3) + 2 = 5.67

count match < treshold, maka score = 0

score = score + Sm*0.5 = 0

score = score + Ord*0.5 = 0

score = score + count_match/W = 5 / 42 = 0.12

Heuristic_score = count_match + 0.5*Sm + 0.5*Ord + count_match/W

= 0 + 0 +0 + 0.12 = 0.12

Passages dengan bobot tertinggi terletak pada kandidat

ID	Dokumen asli	Dokumen Named Entity Tagging	Dokumen preprocessing
37	Semester pendek adalah suatu semester untuk mahasiswa tertentu yang penyelenggaraannya diperpendek dengan mengacu pada ketentuan yang ditetapkan oleh Ketua Program.	Semester pendek adalah <*description*> suatu semester untuk <*organization*>mahasiswa</*organization*> tertentu yang penyelenggaraannya diperpendek dengan mengacu pada ketentuan yang ditetapkan oleh Ketua Program </*description*>	semester pendek semester untuk mahasiswa selenggara pendek kacu tentu yang tetap oleh ketua program

Perolehan Entitas Jawaban

suatu semester untuk mahasiswa tertentu yang penyelenggaraannya diperpendek dengan mengacu pada ketentuan yang ditetapkan oleh Ketua Program

2. Pembobotan Rule-Based

Pertanyaan :

Apa itu semester pendek ?

Preprocessing pertanyaan :

Kata tanya : APA

Keyword : | semester | pendek |

Contoh dokumen sebelum preprocessing :

1. Semester adalah satuan waktu kegiatan yang terdiri atas 16 minggu kuliah atau kegiatan terjadwal lainnya, berikut kegiatan iringannya, termasuk 2 sampai 3 minggu kegiatan penilaian.
2. Semester pendek adalah suatu semester untuk mahasiswa tertentu yang penyelenggaraannya diperpendek dengan mengacu pada ketentuan yang ditetapkan oleh Ketua Progam.
3. Ujian Akhir Semester adalah kegiatan evaluasi yang dilakukan pada akhir semester.
4. Program semester pendek bertujuan memberikan kesempatan kepada mahasiswa untuk memperbaiki nilai mata kuliah yang sudah pernah ditempuh dalam rangka meningkatkan indeks prestasi kumulatif dan memperpendek masa studi serta menghindari terjadinya putus studi.
5. Penyelenggaraan program semester pendek meliputi kegiatan tatap muka, praktikum (bagi mata kuliah berpraktikum), tugas terstruktur, tugas mandiri, ujian tengah dan ujian akhir.
6. Kurikulum dan peraturan akademik dalam perkuliahan semester pendek tetap mengacu pada kurikulum dan peraturan akademik yang berlaku saat itu.
7. Semester pendek tidak diperhitungkan dalam perhitungan masa studi.

Hasil akhir pencarian dokumen setelah preprocessing berdasarkan keyword :

1. semester satu waktu giat atas 16 minggu kuliah giat jadwal ikut giat iringannya masuk 2 3 minggu giat penilaian
2. semester pendek semester untuk mahasiswa selenggara pendek kacu tentu yang tetap oleh ketua progam
3. uji akhir semester giat evaluasi laku akhir semester
4. program semester pendek tuju ikan sempat mahasiswa untuk baik nilai mata kuliah tempuh rangka meningkatkan indeks prestasi kumulatif pendek masa studi serta hindar jadi putus studi
5. selenggara program semester pendek liput giat tatap muka praktikum mata kuliah praktikum tugas struktur tugas mandiri uji tengah uji akhir
6. kurikulum atur akademik kuliah semester pendek tetap kacu kurikulum atur akademik laku saat itu
7. semester pendek hitung hitung masa studi

Ekstraksi Jawaban :

Kandidat	Wordmatch	Rule based APA	Total score
semester satu waktu giat atas 16 minggu kuliah giat jadwal ikut giat iringannya masuk 2 3 minggu giat penilaian	Semester = +3	adalah = +4	7
semester pendek semester untuk mahasiswa selenggara pendek kacu tentu yang tetap oleh ketua progam	Semester = +3 Pendek = +3	adalah = +4	10
uji akhir semester giat evaluasi laku akhir semester	Semester = +3	adalah = +4	7
program semester pendek tuju	Semester = +3	Tidak ada yang	6

ikan sempat mahasiswa untuk baik nilai mata kuliah tempuh rangka meningkatkan indeks prestasi kumulatif pendek masa studi serta hindar jadi putus studi	Pendek = +3	sesuai rule	
selenggara program semester pendek liput giat tatap muka praktikum mata kuliah praktikum tugas struktur tugas mandiri uji tengah uji akhir	Semester = +3 Pendek = +3	Tidak ada yang sesuai rule	6
kurikulum atur akademik kuliah semester pendek tetap kacu kurikulum atur akademik laku saat itu	Semester = +3 Pendek = +3	Tidak ada yang sesuai rule	6
semester pendek hitung hitung masa studi	Semester = +3 Pendek = +3	Tidak ada yang sesuai rule	6

Perolehan Entitas Jawaban dengan skor tertinggi :

Semester pendek adalah suatu semester untuk mahasiswa tertentu yang penyelenggaraannya diperpendek dengan mengacu pada ketentuan yang ditetapkan oleh Ketua Progam.

3.10 Pengujian

Pengujian merupakan tahap dilakukan pengujian berdasarkan implementasi yang telah dibuat melalui pengujian perangkat lunak dengan pengujian akurasi tingkat kebenaran jawaban pada *Question Answering System*.

3.10.1 Perhitungan Presisi, *Recall,* dan *Accuracy*

Pengujian akurasi dilakukan dengan menggunakan 60 sampel pertanyaan dan setiap tipe properti minimal diwakilkan dengan satu pertanyaan. Sampel pertanyaan yang digunakan sebanyak 60 agar dapat mencakup pertanyaan-pertanyaan yang sering ditanyakan mengenai isi buku pedoman. Jawaban yang dikeluarkan sistem dibandingkan dengan jawaban sebenarnya yang sesuai dengan isi Buku Pedoman Pendidikan. Dari hasil pengujian tersebut dihitung nilai presisi dan recall dari pengujian.

Tabel 3.3 Klasifikasi Kelas pada Perhitungan Akurasi, Presisi, dan *Recall*

		Actual Class (Expectation)	
		+	-
Predicted Class (Observation)	+	***TP (True Positive)***	***FP (False Positive)***
	-	***FN (False Negative)***	***TN (True Negative)***

Presisi didapatkan dengan perhitungan sebagai berikut :

$$Precision = \frac{TP}{(TP+FP)} \qquad (3\text{-}1)$$

Recall didapatkan dengan perhitungan sebagai berikut :

$$Recall = \frac{TP}{(TP+FN)} \qquad (3\text{-}2)$$

Accuracy didapatkan dengan perhitungan sebagai berikut :

$$Accuracy = \frac{(TP+TN)}{(TP+FP+TN+FN)} \quad (3\text{-}3)$$

3.10.2 Rancangan Uji Coba

Pengujian sistem dilakukan dengan membandingkan hasil jawaban yang ditemukembalikan oleh sistem berdasarkan pembobotan heuristik Rabin-Karp dan pembobotan Rule-Based. Perbandingan didasarkan pada penilaian sesuai jawaban yang diharapkan, dengan nilai jawaban *TP (True Positive)*, *FP (False Positive)*, *TN (True Negative)*, dan *FN (False Negative)*. Tabel 3.4 adalah tabel perancangan uji coba untuk mengetahui performa dari kedua algoritma.

Tabel 3.4 Perancangan uji coba akurasi

No.	Pertanyaan	Jawaban Yang Diharapkan	Rabin-Karp										Rule-Based	
			Tanpa Stem					Dengan Stem					Tanpa Stem	Dengan Stem
			1	2	3	4	5	1	2	3	4	5		

Sumber : [Rancangan]

Tabel 3.5 Perancangan uji coba kgram

No.	Pertanyaan	Tanpa Stemming		Dengan Stemming	
		Heuristic Score	Waktu Proses	Heuristic Score	Waktu Proses

Sumber : [Rancangan]

Tabel 3.6 Perancangan uji coba modulo

No.	Kgram	Modulo	Tanpa Stemming		Dengan Stemming	
			Heuristic Score	Waktu Proses	Heuristic Score	Waktu Proses

Sumber : [Rancangan]

3.11 Pengambilan Kesimpulan dan saran

Dari hasil pengujian yang telah dilakukan, maka dapat diambil kesimpulan dari hasil implementasi *Question Answering System*. Pengambilan saran diperlukan guna perbaikan dan bahan penunjang bagi penelitian selanjutnya.